Encoder and decoder for image data - has PCM or DPCM processes used to replace dat: using encoding and decoding and for use with CCD line scanners

Bibliographic data | Description | Claims | Mosaics | Original document | INPADOC LEGAL status

Patent number: DE4210246
Publication date: 1993-09-30
Inventor: WIDJAJA HERMAN DR (DE)
Applicant: WIDJAJA HERMAN DR (DE)
Classification:
- international: G06F15/62
- european: H04N7/26C, H04N7/32E, H04N7/34
Application number: DE19924210246 19920328
Priority number(s): DE19924210246 19920328

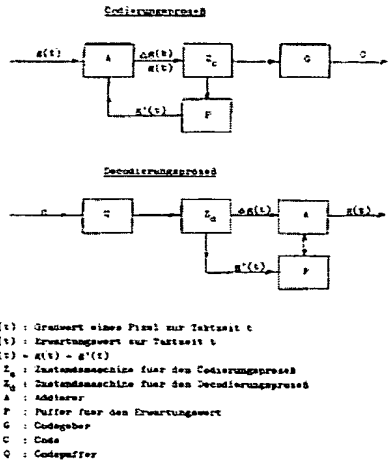
View INPADOC patent family

Abstract of **DE4210246**

The adaptive PCM or DPCM process is used for reducing digital image data. The process is based on encoding process and a decoding process. The encoding process converts data with m bits per pixel into an n bit code, where n is less than m. Dependent upon the determined gray scale difference various code formats may be used i.e. PCM or DPCM. The decoding process serves to generate precise output.

The coding process is based upon 5 functional blocks (ABGDP), with a buffer (P) generating an anticipated value fed back to the input stage.

USE/ADVANTAGE - Suitable for reduction of data from CCD line scanner.



(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Off nl gungsschrift**

(10) **DE 42 10 246 A 1**

(51) Int. Cl.5: G 06 F 15/62

(21) Aktenzeichen: P 42 10 246.4
(22) Anmeldetag: 28. 3. 92
(43) Offenlegungstag: 30. 9. 93

DE 42 10 246 A 1

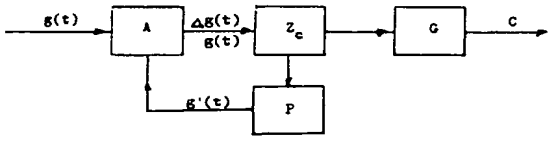
(71) Anmelder:
Widjaja, Herman, Dr., 8000 München, DE

(72) Erfinder:
gleich Anmelder

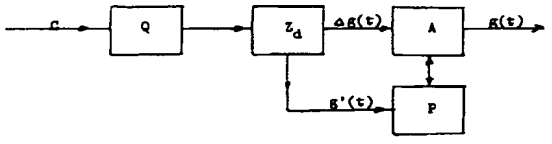
(54) Codierungs- und Decodierungsverfahren sowie -vorrichtungen

(57) Gewöhnliches PCM/DPCM-Verfahren zur Reduktion digitaler Bilddaten hat bekanntlich das folgende Problem: An den Stellen mit großen Grauwertgradienten wie z. B. Kanten, Ecken werden die Grauwerte so stark verfälscht, daß das wiederhergestellte Bild nicht mehr mit dem Urbild übereinstimmt. Ein derart wiederhergestelltes Bild ist insbesondere nicht für die Bildmessung geeignet. Diese Anmeldung stellt ein neues Verfahren vor, das in der Lage ist, das Problem zu lösen und dabei das bewährte Prinzip und die Vorteile der digitalen PCM/DPCM beizubehalten. Der Codierungs- und Decodierungsprozeß ist in der beiliegenden Zeichnung schematisch dargestellt; jeder Prozeß wird als Zustandsmaschine in der Art eines Mealy-Automaten konzipiert. Abhängig von lokalen statistischen Gegebenheiten der Bilddaten kann jedes einzelne Pixel mittels der PCM oder der DPCM in unterschiedlichen Grauwert- und Differenzbereichen codiert werden. Der Decodierungsprozeß stellt die ursprünglichen Bilddaten mit großer Genauigkeit wieder her. Das Verfahren eignet sich besonders gut zur Reduktion von Bilddaten, die mittels eines CCD-Zeilenabtasters erfaßt werden.

Codierungsprozeß



Decodierungsprozeß



g(t) : Grauwert eines Pixel zur Taktzeit t
g'(t) : Erwartungswert zur Taktzeit t
Δg(t) = g(t) - g'(t)
Z_c : Zustandsmaschine fuer den Codierungsprozeß
Z_d : Zustandsmaschine fuer den Decodierungsprozeß
A : Addierer
P : Puffer fuer den Erwartungswert
G : Codegeber
C : Code
Q : Codepuffer

DE 42 10 246 A 1

Die folgenden Angaben sind den vom Anmeld r eingereichten Unterlagen entnommen

BUNDESDRUCKEREI 08. 93 308 039/453

Beschreibung

1. Codierungs- und Decodierungsverfahren sowie -vorrichtungen

Die Erfindung bezieht sich auf ein adaptives PCM/DPCM-Verfahren und ein darauf beruhendes Coder-Decoder-Paar zur Reduktion digitaler Bilddaten. Das Verfahren besteht aus zwei Prozessen: einem Codierungsprozeß und einem Decodierungsprozeß. Der Codierungsprozeß hat die Aufgabe, Bilddaten mit m Bit pro Pixel in n-Bit-Codes umzuwandeln, wobei $n < m$. Abhängig von lokalen statistischen Eigenschaften der Bilddaten wird der Codierungsprozeß mit Hilfe der Pulscode-Modulation (PCM) oder der Differential-Pulscode-Modulation (DPCM) ausgeführt. Der Decodierungsprozeß dient dazu, aus den Codes die ursprünglichen Bilddaten mit fast derselben radiometrischen Genauigkeit wiederherzustellen. Im folgenden werden der Codierungsprozeß und der Decodierungsprozeß zusammen als Prozeß bezeichnet.

Der Coder ist eine Vorrichtung, die den Codierungsprozeß ausführt. Der Decoder ist eine Vorrichtung, die den Decodierungsprozeß ausführt. Der Coder und der Decoder sind als Zustandsmaschinen in der Art eines Mealy-Automaten konzipiert. Jede Zustandsmaschine besteht aus mehreren Zuständen und einer Steuerschaltung, die es der Maschine ermöglicht, von einem Zustand in einen anderen überzugehen. Jeder Zustand stellt einen spezifischen Arbeitsmodus des Prozesses dar.

Die Grauwerte der Bilddaten werden in mehrere PCM-Bereiche unterteilt. Grauwertdifferenzen, sofern sie innerhalb vorgegebener Grenzwerte liegen, werden in mehrere DPCM-Bereiche unterteilt. Ein Zustand wird PCM-Zustand bzw. DPCM-Zustand genannt, wenn ihm ein PCM-Bereich bzw. ein DPCM-Bereich zugeordnet ist. In einem PCM-Zustand wird der Prozeß mit Hilfe der PCM im zugeordneten PCM-Bereich ausgeführt. In einem DPCM-Zustand wird der Prozeß mit Hilfe der DPCM im zugeordneten DPCM-Bereich ausgeführt.

Der Coder und der Decoder haben je einen Anfangszustand, von dem aus der Prozeß im ersten Takt startet. Abhängig vom vorliegenden Pixel (bei der Codierung) oder Code (bei der Decodierung) muß sich der Prozeß entscheiden, ob er zu Beginn des nächsten Takts in einen anderen Zustand übergeht oder in demselben Zustand bleibt. Findet ein Zustandsübergang statt, so gibt der Codierungsprozeß einen n-bit-Code ab, um den Zustandswechsel zu signalisieren. Dieser spezielle Code wird als Steuercode bezeichnet. Beim Decoder veranlaßt der Steuercode den Decodierungsprozeß dazu, in denselben Zustand überzugehen.

Der Prozeß beginnt immer im Anfangszustand, läuft in Intervallen mit einer vorgegebenen Anzahl I von Takten ab und versetzt sich zu Beginn des I-ten Takts in den Anfangszustand zurück. Ein solches Intervall mit I Takten, das im Anfangszustand beginnt und endet, wird Prozeßintervall, kurz Intervall, genannt.

Im Kapitel wird das Verfahren ausführlich beschrieben; anhand eines konkreten Beispieles wird gezeigt, wie der Codierungsprozeß 7-Bit-Bilddaten in 3-Bit-Codes umwandelt und wie der Decodierungsprozeß aus diesen 3-Bit-Codes die ursprünglichen Bilddaten wiederherstellt. Anschließend werden die Blockschaltdiagramme des Coders und des Decoders angegeben und erläutert.

2. Das adaptive PCM/DPCM-Datenreduktionsverfahren

Das Verfahren beruht auf einer wichtigen Eigenschaft digitaler Bilddaten: Sie enthalten sehr viele redundante Informationen, die sich in redundanten Bits ausdrücken lassen. Nahe zueinander liegende Pixel eines Bildes sind meist statistisch miteinander korreliert. Der Grad der Korrelation spiegelt sich in der sogenannten Differenzstatistik wider. Die Differenzstatistik ist die Statistik über die Grauwertdifferenzen von Pixeln, die unmittelbar nebeneinander liegen. Diese Statistik hat im allgemeinen die Form einer Gauß'schen Verteilung mit dem Mittelwert 0 und der Streuung S. Je stärker benachbarte Pixel statistisch miteinander korreliert sind, desto kleiner wird S.

Während diese Eigenschaft die Grundlage für DPCM schafft, bestätigen Ausnahmen in der Praxis die Regel: Die Ausnahmen treten häufig in einem Bild an den Stellen mit großen Grauwertgradienten auf, insbesondere dort, wo Kanten, Kurven, Ecken, Flecke oder sonstige prägnante Texturmerkmale reichlich vorhanden sind. Bei konventionellen PCM/DPCM-Verfahren versucht man, große und abrupte Änderungen der Grauwerte benachbarter Pixel durch PCM abzufangen und dabei das Kompressionsverhältnis konstant zu halten. D. h. der Coder muß sich an solchen Stellen von DPCM auf PCM umstellen und dabei einen Code abgeben, der den Decoder dazu veranlaßt, sich ebenfalls von DPCM auf PCM umzustellen. Um das Kompressionsverhältnis konstant zu halten, muß der Coder in diesem Falle den Code, der eigentlich zur Wiederherstellung eines Pixel benutzt werden soll, für die Umstellung einzusetzen. Da der Decoder nach der Umstellung auf PCM keine Möglichkeit hat, den Grauwert des betroffenen Pixel zu erfahren, ersetzt er den "verlorenen" Grauwert einfach durch den Grauwert des zuletzt wiederhergestellten Pixel. Die Folge: Die Grauwerte an den Stellen mit den prägnanten Texturmerkmalen werden so stark verfälscht, daß das wiederhergestellte Bild nicht für Bildmessung verwendet werden kann.

Die wichtigste Aufgabe des vorliegenden Verfahrens besteht darin, diesen Mangel zu beseitigen und dabei das bewährte Prinzip und die Vorteile der digitalen PCM/DPCM beizubehalten. Um diese Aufgabe zu erfüllen, muß der Coder in der Lage sein, bei Pixeln mit großem Grauwertgradienten jeweils zwei Codes abzugeben: der erste veranlaßt den Decoder dazu, sich auf den entsprechenden Zustand umzustellen; der zweite ermöglicht es ihm, im neuen Zustand das betroffene Pixel wiederherzustellen. Da der erste Code die Aufgabe hat, den Decodierungsprozeß zu steuern, wird er Steuercode genannt. Werden die Bilddaten zeilenweise codiert und sind in jeder Zeile J Pixel enthalten, dann wird die Anzahl L der von einem Coder bei der Codierung einer Bildzeile abgegebenen Codes eine Veränderliche sein, die von lokalen statistischen Eigenschaften der Bilddaten abhängt. Im allgemeinen gilt $L > J$, wobei $(L-J)$ die Anzahl der Steuercodes angibt, die der Coder zur Codierung einer Bildzeile

benötigt. Diese Differenz stellt auch die Anzahl der Zustandsübergänge dar, die der Codierungsprozeß bzw. der Decodierungsprozeß machen muß, um eine Bildzeile mit J Pixeln zu codieren bzw. zu decodieren. Aus diesem Grunde ist das Kompressionsverhältnis bei diesem neuen Verfahren nicht konstant; es hängt von der Differenzstatistik der Bilddaten ab. Je kleiner die Streuung S der Differenzstatistik ist, desto größer wird das Kompressionsverhältnis.

Im folgenden wird anhand eines konkreten Beispiels gezeigt, wie der Codierungsprozeß Bilddaten mit 7 Bit pro Pixel in 3-Bit-Codes umwandelt und wie der Decodierungsprozeß aus diesen Codes die ursprünglichen Bilddaten wiederherstellt. Um die optimale Anzahl von PCM- und DPCM-Zuständen für den Coder/Decoder zu ermitteln, wurde Simulation anhand von Bilddaten mit 7 Bits pro Pixel und 6912 Pixel pro Zeile ausgeführt. Es hat sich dabei herausgestellt, daß es zum besten Kompromiß zwischen dem Kompressionsverhältnis und der Genauigkeit der wiederhergestellten Bilddaten kommt, wenn man bei der Verwendung von 3-Bit-Codes der Coder/Decoder jeweils als Zustandsmaschine mit 3 PCM- und zwei DPCM-Zuständen wie folgt konzipiert:

Zustand	Bereich	Arbeitsmodus
Z0	DPCM1	Grobe DPCM
Z4	DPCM2	Feine DPCM
Z1	PCM1	PCM für den oberen Grauwertbereich
Z2	PCM2	PCM für den mittleren Grauwertbereich
Z3	PCM3	PCM für den unteren Grauwertbereich

Der Automatengraph der Zustandsmaschine ist in **Abb.** 1 angegeben, wobei gerichtete Kanten (Pfeile) die zulässigen Zustandsübergänge darstellen. DPCM1 und DPCM2 werden in der Tabelle 1 dargestellt; PCM1, PCM2 und PCM3 in der Tabelle 2.

Z0 ist der Anfangszustand der Zustandsmaschine, in dem jedes Prozeßintervall beginnt und endet. Z0 ist auch der gemeinsame, nächste Zustand von Z1, Z2 und Z3. D. h. befindet sich ein Prozeß in Z1, Z2 oder Z3, so geht er zu Beginn des nächsten Takts automatisch in Z0 über.

2.1 Der Codierungsprozeß (**Abb.** 2)

Abb. 2 stellt den Codierungsprozeß schematisch dar, der aus 5 Funktionsblöcken A, B, G, D, P besteht. Wir gehen davon aus, daß im Takt t ein Pixel mit dem Grauwert g(t) vorliegt, der Puffer P den Erwartungswert g'(t) enthält, und sich der Codierungsprozeß im Zustand Z(t) befindet. Ist Z(t) gleich Z0 oder Z4, dann errechnet A die Differenz

$$\Delta g(t) = g(t) - g'(t)$$

und leitet sie dem Funktionsblock B zu. Ist Z(t) gleich Z1, Z2 oder Z3, dann liefert A den aktuellen Grauwert g(t) unmittelbar an den Funktionsblock B. B bestimmt anhand von g(t) bzw. Δg(t) den nächsten Zustand Z(t+1). D hat die Aufgaben, den nächsten Erwartungswert g'(t+1) zu ermitteln und g'(t) im Puffer P durch g'(t+1) zu ersetzen. B und D entscheiden gemeinsam darüber, welchen Code der Codegeber G in jedem einzelnen Takt abzugeben hat.

Liegt zu Beginn des Takts t ein Grauwert vor, so hängt das Verhalten des Codierungsprozesses von seinem aktuellen Zustand ab. Es ist deshalb notwendig, den Codierungsprozeß in jedem einzelnen Zustand zu betrachten.

Zustand Z0: Grobe DPCM

Das Verhalten des Codierungsprozesses hängt vom Betrag der Grauwertdifferenz g(t) folgendermaßen ab:

Fall 0.1: $|\Delta g(t)| > 11$

Je nachdem, in welchem PCM-Bereich g(t) liegt, gibt der Codierungsprozeß einen Steuercode C ab und geht zu Beginn des nächsten Takts in einen der 3 PCM-Zustände über:

g(t)	Steuercode	Z(t+1)
127 >= g(t) >= 80	C=5	Z1
80 > g(t) >= 40	C=6	Z2
40 > g(t) >= 0	C=7	Z3

Fall 0.2: $|\Delta g(t)| <= 5$

Der Codierungsprozeß gibt den Steuercode C=0 ab und geht zu Beginn des nächsten Takts in Z4 über.

Fall 0.3: $5 < |\Delta g(t)| <= 11$

Der Codierungsprozeß bleibt in demselben Zustand und gibt einen Code C für die Grauwertdifferen $\Delta g(t)$ ab:

$C = 1$, wenn $-11 <= \Delta g(t) < -8$;
$C = 2$, wenn $-8 <= \Delta g(t) < -5$;
$C = 3$, wenn $5 < \Delta g(t) <= 8$;
$C = 4$, wenn $8 < \Delta g(t) <= 11$.

Zustand Z4: Feine DPCM

Das Verhalten des Codierungsprozesses hängt vom Betrag der Grauwertdifferenz g(t) folgendermaßen ab:

Zustand 4.1: $|\Delta g(t)| > 5$

Der Codierungsprozeß gibt den Steuercode $C = 0$ ab und geht in den Zustand Z0 über.

Fall 4.2: $|\Delta g(t)| <= 5$

Der Codierungsprozeß bleibt in demselben Zustand und gibt einen Code C für die Grauwertdifferenz $\Delta g(t)$ ab:

$C = 1$, wenn $-5 <= \Delta g(t) < -2$;
$C = 2$, wenn $\Delta g(t) = -2$;
$C = 3$, wenn $\Delta g(t) = -1$;
$C = 4$, wenn $\Delta g(t) = 0$;
$C = 5$, wenn $\Delta g(t) = 1$;
$C = 6$, wenn $\Delta g(t) = 2$;
$C = 7$, wenn $2 < \Delta g(t) <= 5$.

Zustand Z1: PCM für den oberen Grauwertbereich

Der Codierungsprozeß ermittelt aus dem Bereich PCM1 (Tabelle 2) den neuen Erwartungswert $g'(t+1)$, gibt dafür einen Code C ab, und versetzt sich zu Beginn des nächsten Takts automatisch in den Anfangszustand Z0 zurück.

g(t)	g'(t+1)	Code
$80 <= g(t) < 86$	82	C = 1
$86 <= g(t) < 92$	88	C = 2
$92 <= g(t) < 98$	94	C = 3
$98 <= g(t) < 104$	100	C = 4
$104 <= g(t) < 110$	106	C = 5
$110 <= g(t) < 116$	112	C = 6
$116 <= g(t) < 122$	118	C = 7
$122 <= g(t) < 128$	124	C = 0

Zustand Z2: PCM für den mittleren Grauwertbereich

Wie Zustand Z1, der neue Erwartungswert $g'(t = 1)$ wird jedoch aus dem Bereich PCM2 (Tabelle 2) ermittelt.

g(t)	g'(t+1)	Code
$40 <= g(t) < 45$	42	C = 1
$45 <= g(t) < 50$	47	C = 2
$50 <= g(t) < 55$	52	C = 3
$55 <= g(t) < 60$	57	C = 4
$60 <= g(t) < 65$	62	C = 5
$65 <= g(t) < 70$	67	C = 6
$70 <= g(t) < 75$	72	C = 7
$75 <= g(t) < 80$	77	C = 0

Zustand Z3: PM für den unteren Grauwertbereich

Wie Zustand Z1, der neue Erwartungswert $g'(t = 1)$ wird jedoch aus dem Bereich PCM3 (Tabelle 2) ermittelt.

g(t)	g'(t + 1)	Code
0 <= g(t) < 5	2	C = 1
5 <= g(t) < 10	7	C = 2
10 <= g(t) < 15	12	C = 3
15 <= g(t) < 20	17	C = 4
20 <= g(t) < 25	22	C = 5
25 <= g(t) < 30	27	C = 6
30 <= g(t) < 35	32	C = 7
35 <= g(t) < 40	37	C = 0

2.2 Der Decodierungsprozeß (**Abb. 3**)

Abb. 3 stellt den Decodierungsprozeß schematisch dar, der aus 5 Funktionsblöcken Q, E, F, S, R besteht. Wir gehen davon aus, daß im Takt t ein Code C vorliegt, der Puffer R den alten Erwartungswert g'(t − 1) enthält und sich der Decodierungsprozess im Zustand Z(t) befindet. Der alte Erwartungswert g'(t − 1) ist im Grunde der Grauwert des zuletzt wiederhergestellten Pixel. Q hat die Aufgabe, den vorliegenden Code C zu überprüfen und im Zusammenhang mit dem aktuellen Zustand Z(t) zu bestimmen, ob ein Pixel wiederhergestellt oder ein neuer Zustand Z(t + 1) ermittelt wird. Ist Z(t) gleich Z0 oder Z4, und es wird ein Pixel wiederhergestellt, so ermittelt E aus der Tabelle 1 eine dem Code C entsprechende Grauwertdifferenz Δg'(t) und leitet sie dem Funktionsblock S zu. S ist ein Addierer, der die Summe

$$g'(t) = g'(t-1) = \Delta g'(t)$$

bildet und sie als Grauwert des wiederherzustellenden Pixel abgibt. Zur gleichen Zeit ersetzt g'(t) den alten Erwartungswert g'(t − 1) in R. Ist Z(t) gleich Z1, Z2 oder Z3, dann ermittelt F aus der Tabelle 2 einen dem Code C entsprechenden Grauwert g'(t) und gibt ihn als Grauwert des wiederherzustellenden Pixel ab. Zur gleichen Zeit ersetzt g'(t) den alten Erwartungswert g'(t − 1) in R. Der Decodierungsprozeß geht zu Beginn des Takts (t + 1) automatisch in den Anfangszustand über.

Liegt zu Beginn des Takts t ein Code vor, so hängt das Verhalten des Decodierungsprozesses von seinem aktuellen Zustand ab. Es ist deshalb notwendig, den Decodierungsprozeß in jedem einzelnen Zustand zu betrachten.

Zustand Z0: Grobe DPCM

Fall 0.1: C = 5, 6, 7 oder 0

Der Decodierungsprozeß braucht nichts anderes zu tun als zu Beginn des nächsten Takts in den dem Steuercode C entsprechenden Zustand überzugehen:

Code	Z(t + 1)
C = 5	Z1
C = 6	Z2
C = 7	Z3
C = 0	Z4

Fall 0.2: C = 1, 2, 3 oder 4

Der Decodierungsprozeß bleibt in demselben Zustand und errechnet g'(t) wie folgt:

$$g'(t) = g'(t-1) + \Delta g'(t)$$

wobei Δg'(t) eine vom Code C abhängige Größe ist, die anhand der Tabelle 1 ermittelt wird:

Code	Δg'(t)
C = 1	−10
C = 2	−7
C = 3	7
C = 4	10

Der Prozeß gibt g'(t) als Grauwert des wiederherzustellenden Pixel ab und ersetzt g'(t − 1) durch g'(t).

Zustand Z4: Feine DPCM

Wenn C=0, dann tut der Decodierungsprozeß nichts anderes als sich zu Beginn des nächsten Takts in den Anfangszustand zurück zu versetzen. Bei allen anderen Codes bleibt der Decodierungsprozeß in demselben Zustand und errechnet g'(t) wie folgt:

$$g'(t) = g'(t-1) + \Delta g'(t)$$

wobei $\Delta g'(t)$ eine vom Code C abhängige Größe ist, die anhand der Tabelle 1 ermittelt wird:

Code	$\Delta g'(t)$
C=1	−4
C=2	−2
C=3	−1
C=4	0
C=5	1
C=6	2
C=7	4

Der Prozeß gibt g'(t) als Grauwert des wiederherzustellenden Pixels ab und ersetzt g'(t−1) durch g'(t).

PCM-Zustand Z1, Z2 oder Z3

Der Decodierungsprozeß gibt einen dem Code C zugeordneten Grauwert g'(t) ab, ersetzt g'(t−1) durch g'(t) und geht zu Beginn des nächsten Takts in den Anfangszustand Z0 über.

Zustand Z1:

PCM für den oberen Grauwertbereich

Code	g'(t)
C=1	82
C=2	88
C=3	94
C=4	100
C=5	106
C=6	112
C=7	118
C=0	124

Zustand Z2:

PCM für den mittleren Grauwertbereich

Code	g'(t)
C=1	42
C=2	47
C=3	52
C=4	57
C=5	62
C=6	67
C=7	72
C=0	77

Zustand Z3:

PCM für den oberen Grauwertbereich

Code	g'(t)
C = 1	2
C = 2	7
C = 3	12
C = 4	17
C = 5	22
C = 6	27
C = 7	32
C = 0	37

2.3 Ein Beispiel

Tabelle 3 zeigt in allen Einzelheiten, wie sich der Coder und der Decoder verhalten, wenn am Eingang des Coders die nachstehende Folge von Grauwerten vorliegt:
10, 13, 18, 22, 84, 83, 86, 85, 84, 86, 48, 53, 55, 58, 57, 50, 56, 54, 52, 49

Der Coder gibt für diese kurze Folge von 20 7-Bit-Pixeln insgesamt 29 3-Bit-Codes ab. Das Kompressionsverhältnis beträgt in diesem Falle:

$$\frac{20 \times 7}{29 \times 3} = \frac{140}{87} = 1{,}61$$

Die mittlere Abweichung der Grauwerte der wiederhergestellten Pixel ist 0,77.

2.4. Fehlerbehandlung

Der Weg, der die Codes vom Ausgang des Coders zum Eingang des Decoders führt, ist leider nicht frei von Störungen. Dieser Weg kann z. B. aus einem Übertragungskanal und/oder einem Datenträger bestehen. Ein fehlerhafter Code kann dazu führen, daß die Zustandsübergänge des Decodierungsprozesses nicht mehr mit den des Codierungsprozesses übereinstimmen. Dies hat zwei Konsequenzen:

- Fehlerhafte Pixel werden hergestellt, die sich auf der ganzen Zeile des Bildes fortpflanzen.
- Die Anzahl der wiederhergestellten Pixel pro Zeile stimmt nicht mit der Zeilenlänge des ursprünglichen Bildes überein.

Man benötigt also zwei Maßnahmen, um das Coder-Decoder-Paar fehlertolerant zu machen:

2.4.1. Prozeß-Intervall

Der Codierungsprozeß und der Decodierungsprozeß werden nach Ablauf einer vorgegebenen Anzahl I von Takten in ihren Anfangszustand zurückversetzt. I wird Prozeß-Intervall, kurz Intervall, genannt. Nach der Abgabe (bei der Codierung) bzw. Überprüfung (bei der Decodierung) des Kontrollcodes beginnt das nächste Intervall, indem der Prozeß erneut im Anfangszustand startet. Tritt beim Decodierungsprozeß ein fehlerhafter Code auf, so kann sich der Fehler nur bis zum Ende des Intervalls fortpflanzen. Auf diese Weise wird die Fortpflanzung fehlerhafter Pixel auf höchstens ein Intervall beschränkt. Andererseits muß man berücksichtigen, daß jedes Intervall im Durchschnitt zwei zusätzliche Zustandsübergänge, d. h. zwei zusätzliche Codes, benötigt. Ein allzu kurzes Intervall kann also dazu führen, daß sich das Kompressionsverhältnis wesentlich verschlechtert. Ein geeignetes Intervall soll stets ein Kompromiß zwischen zwei entgegengesetzten Anforderungen sein: Die Fehlerfortpflanzung möglichst schnell zu stoppen und ein hohes Kompressionsverhältnis zu erzielen. Simulation anhand von 7-Bit-bilddaten (Zeilenlänge: 6912 Pixel) hat gezeigt, daß ein guter Kompromiß in einem Intervall von 26—30 Takten zu finden ist.

2.4.2. Kontrollcode

Im Normalbetrieb muß die Anzahl K der innerhalb eines Intervalls vom Decoder wiederhergestellten Pixel identisch sein mit der Anzahl J der vom Coder in demselben Intervall codierten Pixel. Ein fehlerhafter Code kann jedoch dazu führen, daß $J > K$ oder $J < K$. Um dies zu kontrollieren, gibt der Coder am Ende eines jeden Intervalls einen zusätzlichen Code aus, der die Differenz $D = I - J$ kennzeichnet. Dabei ist I eine Konstante, die das Intervall angibt. Am Ende eines jeden Int rvalls überprüft der Decoder den Kontrollcode, stellt $J = I - D$ wieder her und vergleicht J mit K. Hierzu gibt es drei Möglichkeiten:

J = K: Kein Fehler. Der Dekodierungsprozeß setzt sich im nächsten Intervall fort.

J > K: Zuwenig Pixel wurden in diesem Intervall wiederhergestellt. Der Decodierungsprozeß ergänzt die Grauwerte der fehlenden (J − K) Pixel durch den Grauwert des zuletzt im Intervall wiederhergestelltem Pixel. Der Prozeß zeigt den Fehler an und setzt sich im nächsten Intervall fort.

J < K: Zuviel Pixel wurden in diesem Intervall wiederhergestellt. Der Decodierungsprozeß vernachlässigt die letzten (K − J) Pixel, zeigt den Fehler an und setzt sich im nächsten Intervall fort.

Wenn eine Bildzeile vollständig wiederhergestellt ist, wird die Fehleranzeige geprüft. Ist die Anzeige gesetzt, so wird sie zurückgesetzt und die entsprechende Zeilennummer in einen dafür vorgesehenen Puffer eingetragen.

3. Der Coder und der Decoder

Die Blockschaltdiagramme des Coders und des Decoders sind in **Abb.** 4 und **Abb.** 5 dargestellt.

3.1 Der Coder (**Abb.** 4)

Gehen wir davon aus, daß sich der Coder zu Beginn des Takts t im Zustand Z(t) befindet und der Puffer **43** den Erwartungswert g'(t) enthält. Liegt zu diesem Zeitpunkt ein neuer Grauwert g(t) vor, so errechnet der Volladdierer **41** die Differenz

$$\Delta g(t) = g(t) - g'(t)$$

und leitet sie dem Ergebnisregister **42** zu. Das Ergebnis ermöglicht es dem Adressengenerator **44**, den nächsten Zustand Z(t + 1) zu ermitteln.

Fall 1: Z(t + 1) ist nicht gleich Z(t)

Ein Zustandsübergang findet statt und der Coder muß einen Steuercode abgeben. Der Adressengenerator **46** ermittelt aus der Registerdatei **47** einen dem Zustandsübergang entsprechenden Steuercode und liefert ihn an den Codepuffer **4C**.

Fall 2: Z(t + 1) ist gleich Z(t)

Der Coder bleibt in demselben Zustand. Hierin gibt es zwei Möglichkeiten:

Fall 2.1: (Z(t + 1) ist ein DPCM-Zustand (z. B. Z0 oder Z4)

Liegt die aktuelle Differenz Δg(t) vor, dann ermittelt der Adressengenerator **48** aus der Registerdatei **49** die quantisierte Differenz Δg'(t) und leitet sie dem Puffer **43** zu. Der Puffer **43** enthält nun zwei Werte: g'(t) und Δg'(t). Der Volladdierer **41** errechnet aus diesen Werten einen neuen Erwartungswert

$$g'(t+1) = g'(t) = \Delta g'(t),$$

legt ihn im Ergebnisregister **42** ab und ersetzt g'(t) im Puffer **43** durch diesen neuen Erwartungswert. Gleichzeitig ermittelt der Adressengenerator **46** aus der Registerdatei **47** einen dem neuen Erwartungswert entsprechenden Code und liefert ihn an den Codepuffer **4C**.

Fall 2.2: Z(t + 1) ist ein PCM-Zustand (z. B. Z1, Z2 oder Z3)

Der aktuelle Grauwert g(t) wird durch das Ergebnisregister **42** dem Adressengenerator **48** vorgelegt. Dieser ermittelt aus der Registerdatei **49** den entsprechenden quantisierten Grauwert, der als neuer Erwartungswert g'(t + 1) dem Puffer **43** und dem Adressengenerator **46** zugeleitet wird. Der Adressengenerator **46** sucht aus der Registerdatei **47** einen dem Erwartungswert g'(t + 1) entsprechenden Code aus und legt ihn im Codepuffer **4C** ab.

Der Takt-und-Pixel-Zähler **4A** hat die Aufgabe, die Differenz zwischen der Anzahl der Takte und der Anzahl der codierten Pixel in einem jeden Intervall zu bilden. Am Ende eines jeden Intervalls gibt der Kontrollcode-Generator **4B** einen Kontrollcode ab, der dieser Differenz entspricht.

3.2 Der Decoder (**Abb.** 5)

Der Adressengenerator **52** überprüft jeden beim Codepuffer **50** eintreffenden Code und entscheidet zunächst, ob der Decoder eine Zustandsänderung vorzunehmen hat. Ist dies der Fall, so bestimmt der Adressengenerator **52** mit Hilfe des Zustandsregisters **51** den neuen Zustand und veranlaßt den Decoder dazu, zu Beginn des nächsten Takts in den neuen Zustand überzugehen. Andernfalls wird keine Zustandsänderung in Gang gesetzt und, je nachdem in welchem Zustand sich der Decoder gerade befindet, ermittelt der Adressengenerator **52** aus der Registerdatei **53** einen dem vorliegenden Code entsprechenden Grauwert g'(t) (PCM-Zustand) oder eine

dem vorliegenden Code entsprechende Grauwertdifferenz $\Delta g'(t)$ (DPCM-Zustand). Ist der Decoder in einem PCM-Zustand, so wird $g'(t)$ als nächster Erwartungswert im Puffer **54** zwischengespeichert und als Grauwert des wiederherzustellenden Pixel dem Pixelpuffer **57** zugeleitet. Ist der Decoder in einem DPCM-Zustand, so wird $\Delta g'(t)$ dem Puffer **56** zugeführt. Zu diesem Zeitpunkt enthält der Puffer **54** noch den alten Erwartungswert $g'(t-1)$. Der Volladdierer **55** bildet die Summe $g'(t)$

$$g'(t) = g'(t-1) + \Delta g'(t)$$

und liefert sie als Grauwert des wiederherzustellenden Pixel an den Pixelpuffer **57**. Gleichzeitig wird $g'(t)$ als nächster Erwartungswert im Puffer **54** zwischengespeichert.

Trifft ein Kontrollcode am Endes des Intervalls beim Codepuffer **50** ein, so wird dieser der Kontrollcode-Prüfschaltung **59** zugeführt. Diese Prüfschaltung prüft anhand des Kontrollcodes und des Ergebnisses des Pixel-und-Takt-Zählers nach, ob eine Korrektur der wiederhergestellten, noch im Pixelpuffer **57** anstehenden Pixel notwendig ist. Ist dies der Fall, so werden sie nach den in Abschnitt 2.4.2 angegebenen Vorschriften korrigiert.

Tabelle 1

Quantisierung und Codierung von Grauwertdifferenzen im DPCM-Zustand

	Zustand Z0 Grobe DPCM			Zustand Z4 Feine DPCM							Zustand Z0 Grobe DPCM		
Δg:	-11,-10,-9	-8,-7,-6		-5,-4,-3	-2	-1	0	1	2	3,4,5	6,7,8		9,10,11
$\Delta g'$:	-10	-7		-4	-2	-1	0	1	2	4	7		10
Code:	1	2		1	2	3	4	5	6	7	3		4

Δg : Errechnete Grauwertdifferenz
$\Delta g'$: Quantisierte Grauwertdifferenz

Tabelle 2

Quantisierung und Codierung von Grauwerten im PCM-Zustand

Zustand Z1 (PCM1)

g:	127,–,122	121,–,116	115,–,110	109,–,104	103,–,98	97,–,92	91,–,86	85,–,80
g':	124	118	112	106	100	94	88	82
Code:	0	7	6	5	4	3	2	1

Zustand Z2 (PCM2)

g:	79,–,75	74,–,70	69,–,65	64,–,60	59,–,55	54,–,50	49,–,45	44,–,40
g':	77	72	67	62	57	52	47	42
Code:	0	7	6	5	4	3	2	1

Zustand Z3 (PCM3)

g:	39,–,35	34,–,30	29,–,25	24,–,20	19,–,15	14,–,10	9,–,5	4,–,0
g':	37	32	27	22	17	12	7	2
Code:	0	7	6	5	4	3	2	1

g : Urspruenglicher Grauwert
g': Quantisierter Grauwert

Tabelle 3

Einzelheiten des Beispiels 3.3

	Coder			Decoder				
Takt t	g	Z(t)	Z(t+1)	Code	Z(t)	Z(t+1)	g'	g − g'
1	10	Z0	Z0	4	Z0	Z0	10	0
2	13	Z0	Z4	0	Z0	Z4		
3		Z4	Z4	7	Z4	Z4	14	−1
4	18	Z4	Z4	7	Z4	Z4	18	0
5	22	Z4	Z4	7	Z4	Z4	22	0
6	84	Z4	Z0	0	Z4	Z0		
7		Z0	Z1	5	Z0	Z1		
8		Z1	Z0	1	Z1	Z0	82	2
9	83	Z0	Z4	0	Z0	Z4		
10		Z4	Z4	5	Z4	Z4	83	0
11	86	Z4	Z4	7	Z4	Z4	87	−1
12	85	Z4	Z4	2	Z4	Z4	85	0
13	84	Z4	Z4	3	Z4	Z4	84	0
14	86	Z4	Z4	6	Z4	Z4	86	0
15	48	Z4	Z0	0	Z4	Z0		
16		Z0	Z2	6	Z0	Z2		
17		Z2	Z0	2	Z2	Z0	47	1
18	53	Z0	Z0	3	Z0	Z0	54	−1
19	55	Z0	Z4	0	Z0	Z4		
20		Z4	Z4	5	Z4	Z4	55	0
21	58	Z4	Z4	7	Z4	Z4	59	−1
22	57	Z4	Z4	2	Z4	Z4	57	0
23	50	Z4	Z0	0	Z4	Z0		
24		Z0	Z0	2	Z0	Z0	50	0
25	56	Z0	Z0	3	Z0	Z0	57	−1
26	54	Z0	Z4	0	Z0	Z4		
27		Z4	Z4	1	Z4	Z4	53	1
28	52	Z4	Z4	3	Z4	Z4	52	0
29	49	Z4	Z4	1	Z4	Z4	48	1

g: Aktueller Grauwert am Eingang des Coders
g': Wiederhergestellter Grauwert am Ausgang des Decoders
Z(t): Aktueller Zustand
Z(t+1): Naechster Zustand

Patentansprüche

1. Codierungsprozeß wie im ersten Kapitel definiert, der durch folgende Merkmale **gekennzeichnet ist**:
   1.1 Der Anfangszustand ist ein DPCM-Zustand, dem der gröbste DPCM-Bereich, d. h. DPCM-Bereich mit den gröbsten Grauwertdifferenzen, zugeordnet ist.
   1.2 Ist der Codierungsprozeß im Anfangszustand und es liegt ein Grauwert vor, so kann er abhängig von der errechneten Grauwertdifferenz entweder
      a. in einen PCM-Zustand übergehen und dabei einen Steuercode abgeben, oder
      b. in einen anderen DPCM-Zustand übergehen und dabei einen Steuercode abgeben, oder
      c. in demselben Zustand bleiben und dabei einen Code für die aktuelle Grauwertdifferenz abgeben.
   1.3 Ist der Codierungsprozeß in einem PCM-Zustand, so gibt er einen Code für den quantisierten Grauwert des vorliegenden Pixel ab und versetzt sich zu Beginn des nächsten Takts automatisch in den Anfangszustand zurück, ohne dabei einen Steuercode abzugeben.
   1.4 Ist der Codierungsprozeß in einem DPCM-Zustand, der nicht Anfangszustand ist, dann kann er sich abhängig von errechneter Grauwertdifferenz entweder
      a. in demselben Zustand aufhalten und dabei einen Code für die Grauwertdiferenz abgeben, oder
      b. in einen anderen DPCM-Zustand — der auch Anfangszustand sein kann — versetzen und dabei einen Steuercode abgeben.
   1.5 Am Ende eines jeden Prozeßintervalls mit I Takten versetzt sich der Codierungsprozeß in den Anfangszustand zurück, ermittelt die Anzahl J der codierten Pixel und gibt einen Kontrollcode für die

Differenz (I—J) ab. Beim Decoder ermöglicht es dieser Kontrollcode dem Decodierungsprozeß, etwaige Fehler festzustellen und Fehlerfortpflanzung zu stoppen.

2. Decodierungsprozeß wie im ersten Kapitel definiert, der durch folgende Merkmale gekennzeichnet ist:

2.1 Der Anfangszustand ist ein DPCM-Zustand, wie unter Punkt 1.1 angegeben

2.2 Ist der Decodierungsprozeß im Anfangszustand, so hängt sein Verhalten vom vorliegenden Code ab:

— Ist der Code ein Steuercode, so geht der Prozeß entweder in einen PCM-Zustand oder einen anderen DPCM-Zustand über, ohne dabei ein Pixel wiederherzustellen.

— Ist der Code kein Steuercode, so bleibt der Prozeß in demselben Zustand, errechnet aus dem Code einen Grauwert, und stellt das nächste Pixel wieder her.

2.3 Ist der Decodierungsprozeß in einem PCM-Zustand und es liegt ein Code vor, so ermittelt er einen dem Code zugeordneten Grauwert und stellt das nächste Pixel wieder her. Der Prozeß versetzt sich zu Beginn des nächsten Takts automatisch in den Anfangszustand zurück.

2.4 Befindet sich der Decodierungsprozeß in einem DPCM-Zustand, der nicht Anfangszustand ist, so hängt sein Verhalten vom vorliegenden Code ab:

— Ist der Code ein Steuercode, dann versetzt sich der Prozeß in einen anderen DPCM-Zustand — der auch Anfangszustand sein kann — ohne dabei ein Pixel wiederherzustellen.

— Ist der Code kein Steuercode, dann bleibt der Prozeß in demselben Zustand, errechnet aus dem Code einen Grauwert, und stellt das nächste Pixel wieder her.

2.5 Am Ende eines jeden Prozeßintervalls versetzt sich der Decodierungsprozeß in den Anfangszustand zurück, ermittelt die Anzahl K der wiederhergestellten Pixel, und vergleicht sie mit der durch den Kontrollcode übermittelten Anzahl J der im Coder während desselben Intervalls codierten Pixel. Stimmen J und K nicht überein, so sind entweder zuviel oder zuwenig Pixel wiederhergestellt worden. Im ersten Falle werden die überzähligen Pixel vernachlässigt. Im zweiten Falle werden fehlende Pixel durch das im Prozeßintervall zuletzt wiederhergestellte Pixel ergänzt.

3. Coder (**Abb. 4**) wie im ersten Kapitel definiert, dadurch gekennzeichnet, daß er als Zustandsmaschine in der Art eines Mealy-Automaten konzipiert ist und folgende Komponenten aufweist (**Abb. 4**):

— Eine Steuerschaltung (**41, 42, 43, 44, 45**), die aus einem Volladdierer (**41**), einem Ergebnisregister (**42**), einem Puffer (**43**) für quantisierte Grauwerte und Grauwertdifferenzen, einem Adressengenerator (**44**) und einem Zustandsregister (**45**) besteht. Diese Steuerschaltung ermöglicht es dem Coder, in jedem Takt den nächsten Zustand der Maschine zu bestimmten.

— Ein Pixelpuffer (**40**), der dazu dient, einlaufende Pixel aufzufangen.

— Eine Registerdatei (**49**), die quantisierte Grauwerte und Grauwertdifferenzen enthält. Aus dieser Registerdatei können einzelne Werte mit Hilfe des Adressengenerators (**48**) geholt und im Puffer (**43**) abgelegt werden.

— Eine Registerdatei (**47**), in der n-Bit-Codes gespeichert sind. Die einzelnen Codes können mit Hilfe des Adressengenerators (**46**) geholt und im Codepuffer (**4C**) abgelegt werden.

— Ein Pixel-und-Takt-Zähler (**4A**), der die codierten Pixel und die Takte in einem jeden Intervall abzählt.

— Ein Kontrollcode-Generator (**4B**), der am Ende eines jeden Intervalls anhand des Zählergebnisses des Pixel-und-Takt-Zählers einen Kontrollcode bildet und abgibt.

— Ein Codepuffer (**4C**), der dazu dient, die Codes aufzunehmen und sie in vorgeschriebenem Format auszugeben.

4. Decoder (**Abb. 5**) wie im ersten Kapitel definiert, dadurch gekennzeichnet, daß er als Zustandsmaschine in der Art eines Mealy-Automaten konzipiert ist und folgende Komponenten aufweist (**Abb. 5**):

— Ein Codepuffer (**50**), der dazu dient, einlaufende Codes aufzufangen.

— Eine Registerdatei (**53**), in der Erwartungswerte und quantisierte Grauwertdifferenzen gespeichert sind.

— Eine Steuerschaltung (**51, 52**), die aus einem Zustandsregister (**51**) und einem Adressengenerator (**52**) besteht. Das Zustandsregister gibt den aktuellen Zustand der Maschine an. Anhand des vorliegenden Codes bestimmt die Steuerschaltung den nächsten Zustand und ermittelt aus der Registerdatei (**53**) entweder einen Erwartungswert oder eine Grauwertdifferenz.

— Eine Addiererschaltung (**54, 55, 56**), die aus einem Erwartungwert im Puffer (**54**) und einer Grauwertdifferenz im Puffer (**56**) einen neuen Erwartungswert errechnet und das entsprechende Pixel wiederherstellt.

— Ein Pixelpuffer (**57**) zur Aufnahme der in einem Intervall wiederhergestellten Pixel.

— Ein Pixel-und-Takt-Zähler (**58**), der die im Pixelpuffer (**57**) anstehenden, wiederhergestellten Pixel abzählt.

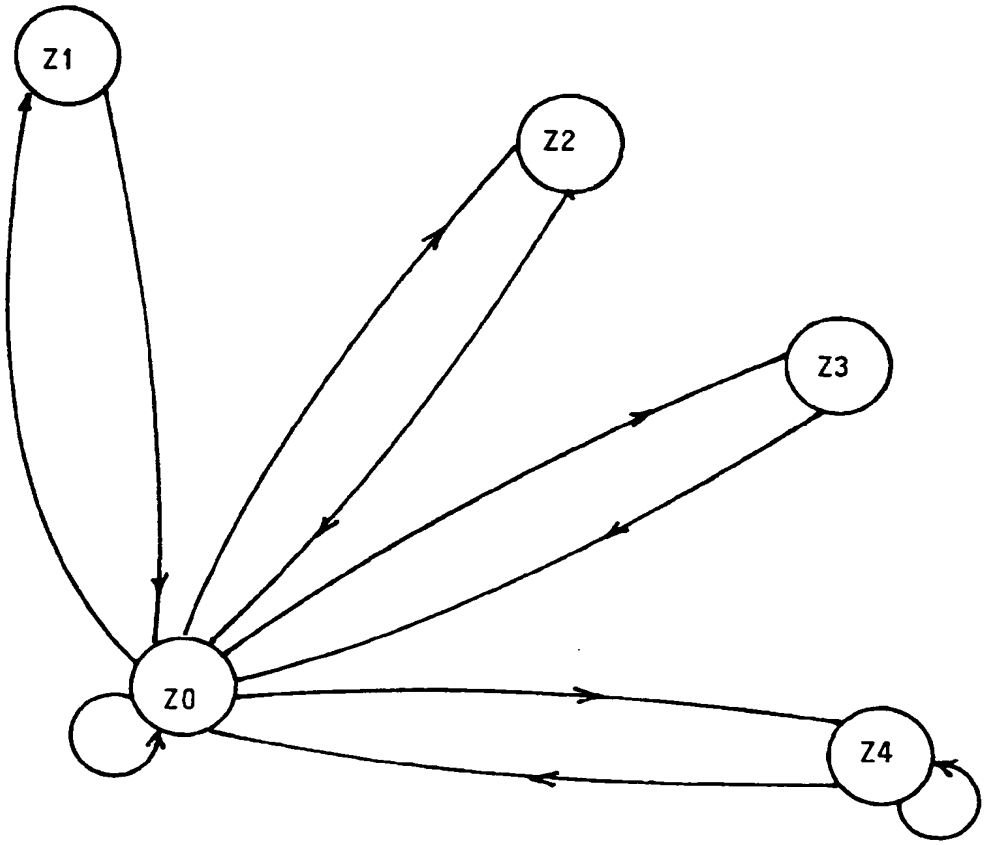
— Eine Kontrollcode-Prüfschaltung (**59**), die am Ende eines jeden Prozeßintervalls prüft, ob die durch den Kontrollcode übermittelte Anzahl J der codierten Pixel mit der Anzahl K der im Intervall wiederhergestellten, noch im Pixelpuffer (**57**) anstehenden Pixel übereinstimmt. Stimmen J und K nicht überein, dann werden die im Pixelpuffer anstehenden Pixel nach dem in Abschnitt 2.4.2 beschriebenen Verfahren korrigiert.

Hierzu 4 Seite(n) Zeichnungen

Nummer: DE 42 10 246 A1
Int. Cl.[5]: G 06 F 15/62
Offenlegungstag: 30. Sept mber 1993

Abbildung 1

Der Automatengraph



Nummer: DE 42 10 246 A1
Int. Cl.[5]: G 06 F 15/62
Offenlegungstag: 30. Sept mber 1993

Abbildung 2

Der Codierungsprozess

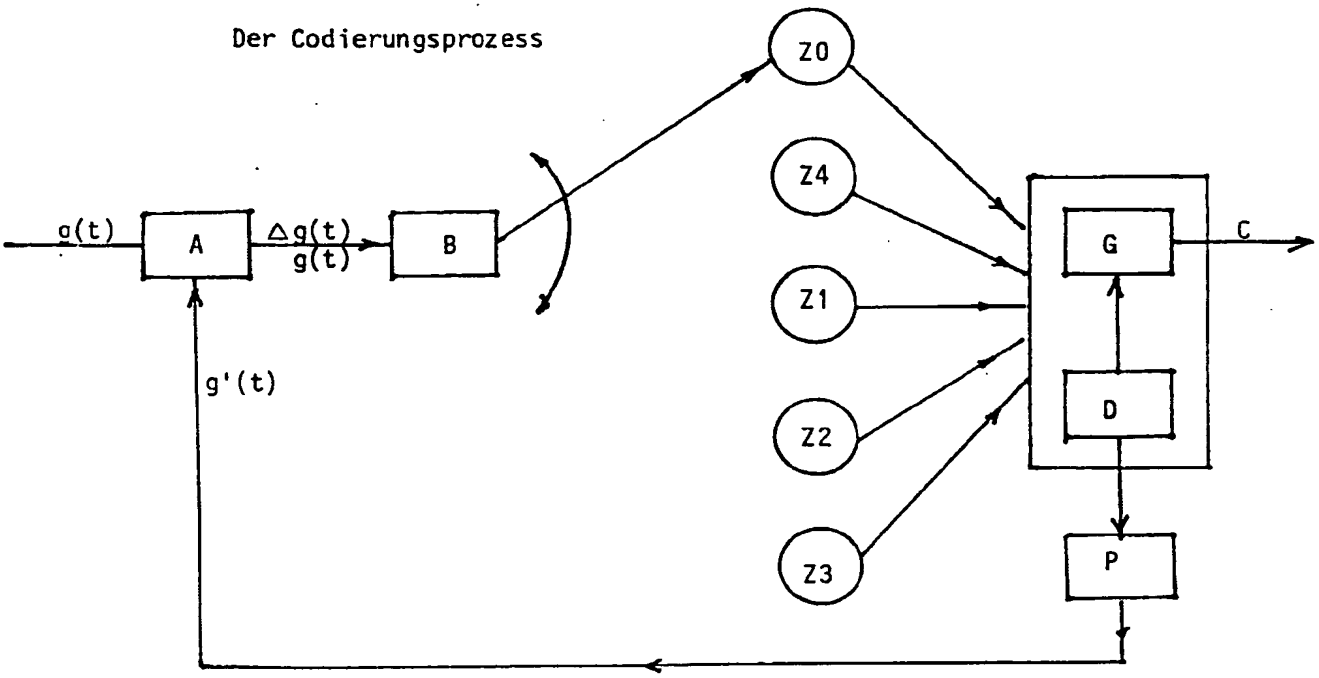
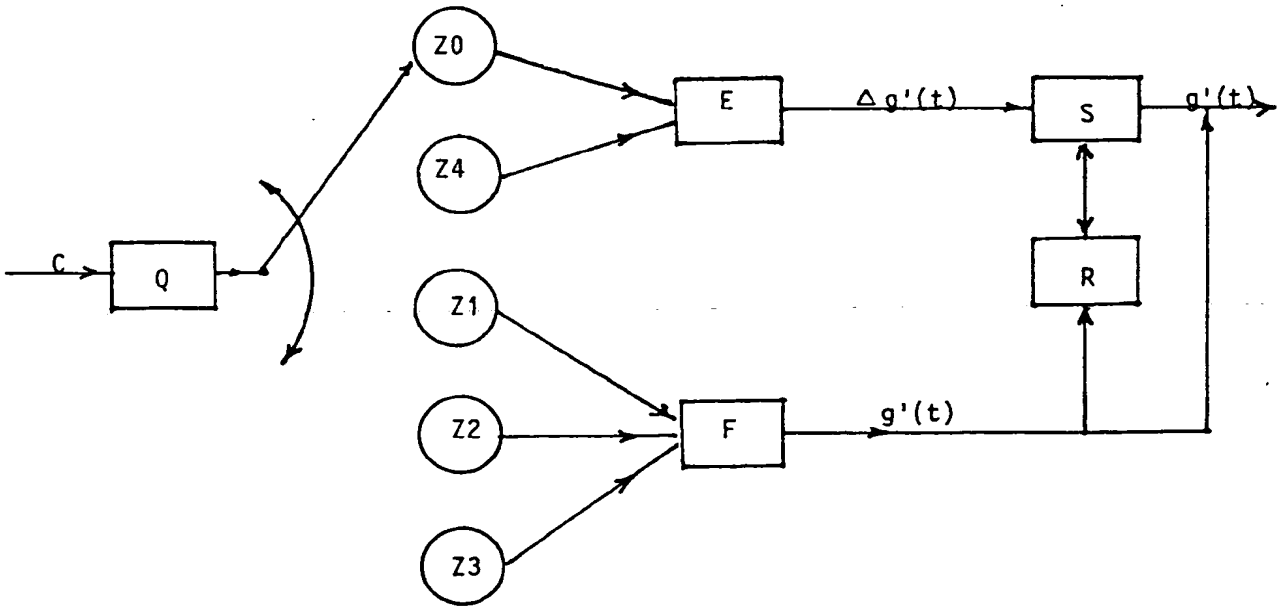


Abbildung 3

Der Decodierungsprozess



Nummer: DE 42 10 246 A1
Int. Cl.[5]: G 06 F 15/62
Offenlegungstag: 30. September 1993

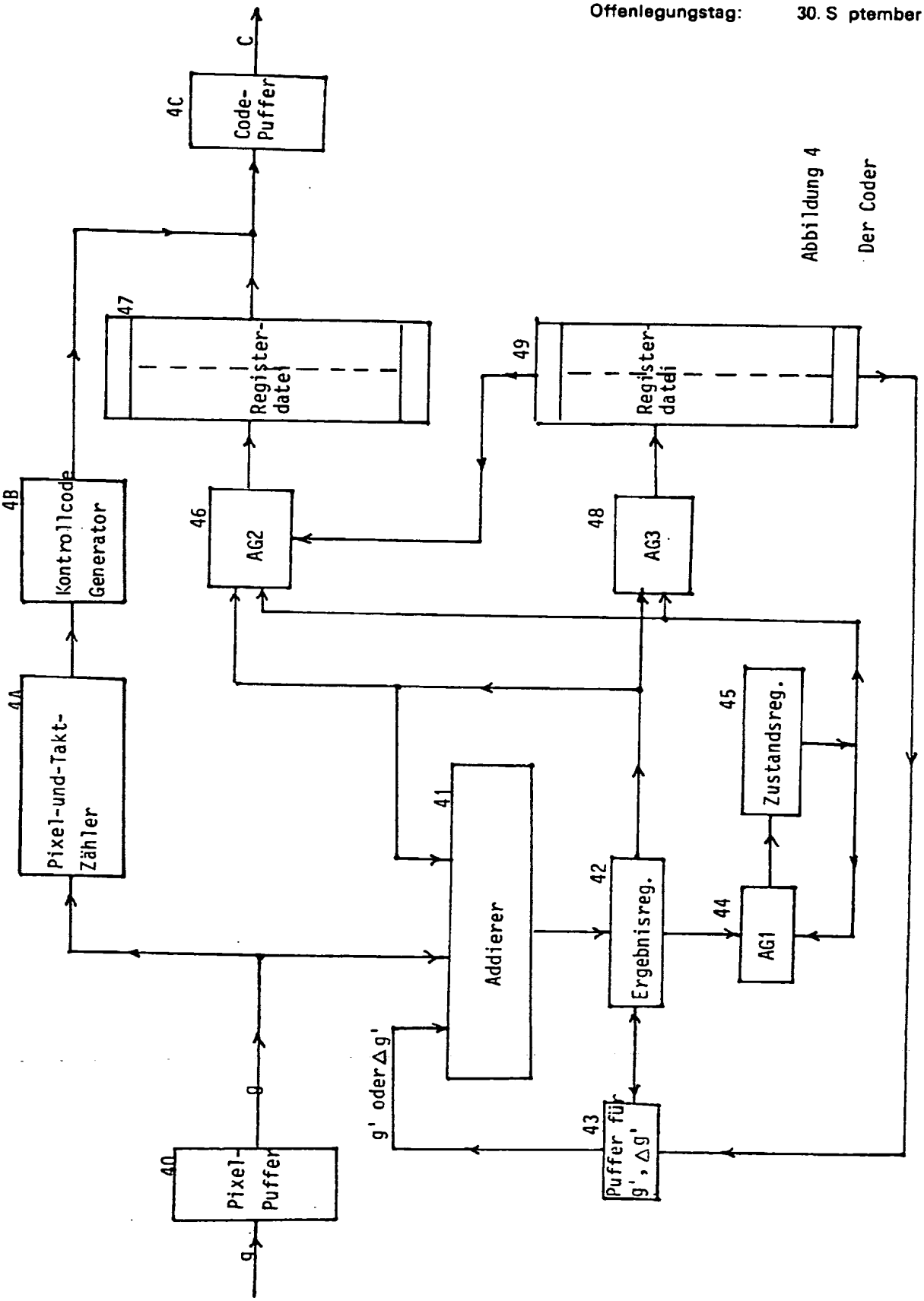


Abbildung 4

Der Coder

Nummer: DE 42 10 246 A1
Int. Cl.[5]: G 06 F 15/62
Offenlegungstag: 30. September 1993

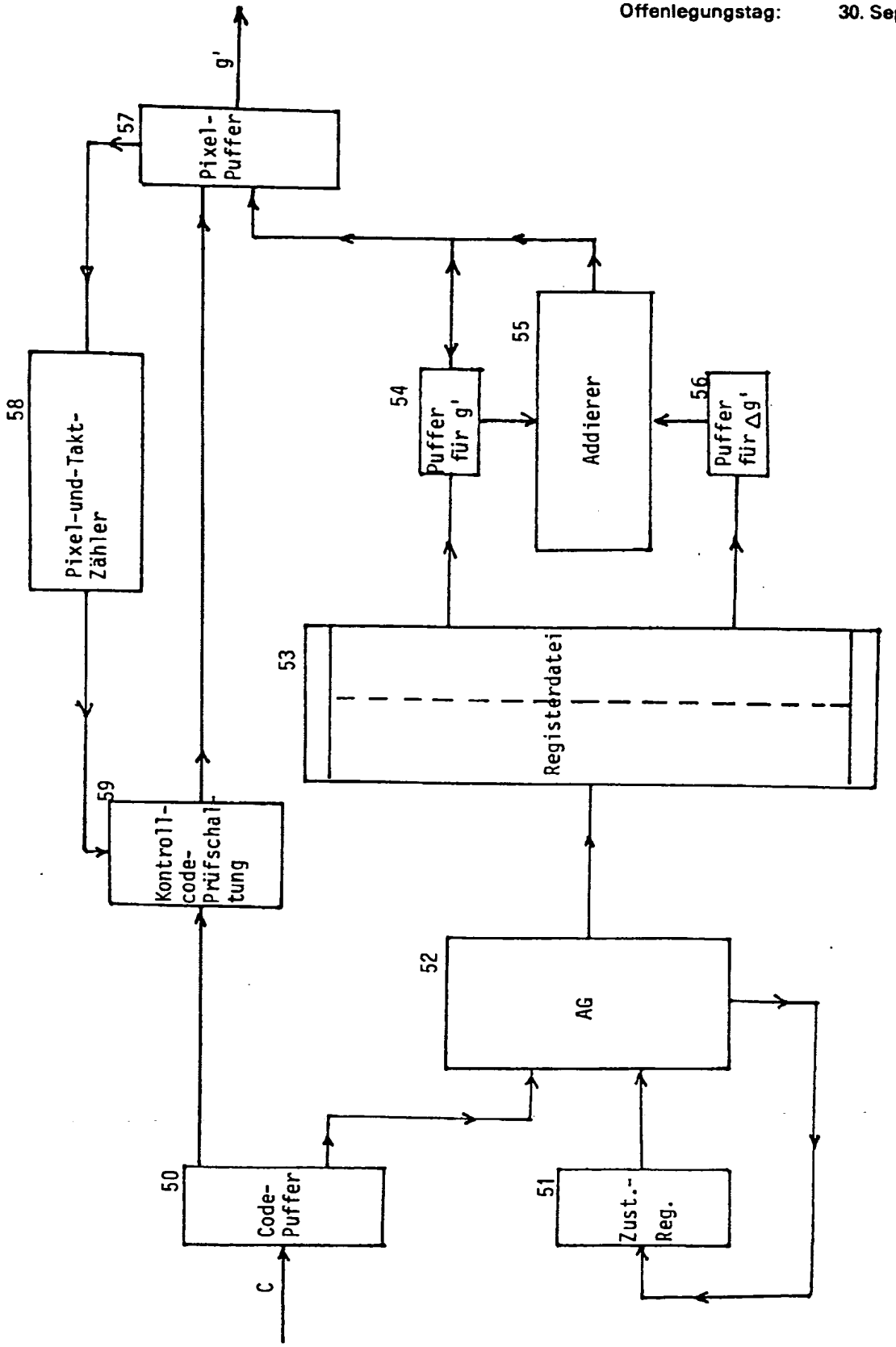


Abbildung 5

Der Decoder